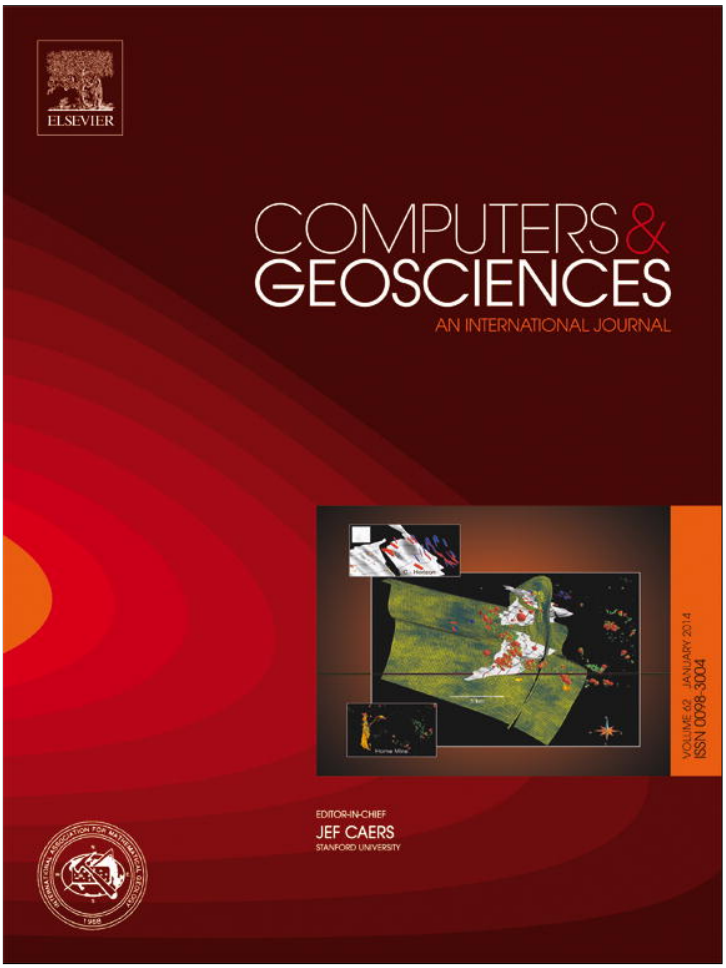

Computers & Geosciences 62 (2014) 168–177

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

CrusDe: A plug-in based simulation framework for composable Crustal Deformation studies using Green's functions

R. Grapenthin [a,b,c,*]

[a] Department of Computer Science, Humboldt-Universität zu Berlin, Berlin, Germany
[b] Geophysical Institute, University of Alaska Fairbanks, Fairbanks, USA
[c] Berkeley Seismological Laboratory, University of California, Berkeley, 209 McCone Hall, Berkeley CA-94720, USA

ARTICLE INFO

Article history:
Received 23 May 2013
Received in revised form
2 July 2013
Accepted 9 July 2013
Available online 23 July 2013

Keywords:
Modeling
Simulation
Crustal deformation
Green's function
Loading
Surface displacement
C
C++

ABSTRACT

CrusDe is a plug-in based simulation framework written in C/C++ for Linux platforms (installation information, download and test cases: http://www.grapenthin.org/crusde). It utilizes Green's functions for simulations of the Earth's response to changes in surface loads. Such changes could involve, for example, melting glaciers, oscillating snow loads, or lava flow emplacement. The focus in the simulation could be the response of the Earth's crust in terms of stress changes, changes in strain rates, or simply uplift or subsidence and the respective horizontal displacements of the crust (over time).

Rather than implementing a variety of specific models, CrusDe approaches crustal deformation problems from a general formulation in which model elements (Green's function, load function, relaxation function, load history), operators, pre- and postprocessors, as well as input and output routines are independent, exchangeable, and reusable on the basis of a plug-in approach (shared libraries loaded at runtime). We derive the general formulation CrusDe is based on, describe its architecture and use, and demonstrate its capabilities in a test case.

With CrusDe users can: (1) dynamically select software components to participate in a simulation (through XML experiment definitions), (2) extend the framework independently with new software components and reuse existing ones, and (3) exchange software components and experiment definitions with other users.

CrusDe's plug-in mechanism aims for straightforward extendability allowing modelers to add new Earth models/response functions. Current Green's function implementations include surface displacements due to the elastic response, final relaxed response, and pure thick plate response for a flat Earth. These can be combined to express exponential decay from elastic to final relaxed response, displacement rates due to one or multiple disks, irregular loads, or a combination of these. Each load can have its own load history and crustal decay function.

1. Introduction and motivation

Crustal deformation studies are concerned with responses of the Earth's lithosphere to endogen (e.g., tectonic or magmatic) and exogen (e.g., glacial) processes. These processes can be understood as a load force that is applied to either the interior or the surface of the lithosphere which is composed of crust and upper mantle. Here, we focus on surface loads which disturb the isostatic equilibrium of the lithosphere. The resulting changes (e.g., uplift or subsidence) can be quantified by geophysical data, which in turn can be modeled to derive properties of the Earth.

To illustrate this, Fig. 1 depicts the Earth–load response system. The surface load and its temporal change, the load history, are the input signal that transform the lithosphere. Those measurable changes are the output signal (e.g. surface displacement, gravity change, etc.). The lithosphere can be understood as a low-pass filter that passes long wavelengths (e.g., Watts, 2001), which means that its properties determine how well the frequency content of the input is reflected in the output. A very flexible crust deforms such that many of the irregularities (small wavelength, high frequency) of an input load (say, a lava flow) can be identified in the output, whereas a stiff crust results in a very smooth gradient over the surface displacements with "smooth edges" (long wavelengths/small frequency). This view provides intuition useful for the rest of the paper: The application of a different filter function only requires swapping out one component of the Earth–load response-system, rather than setting it up from scratch. Amongst others, these filters could express load induced changes in stress state (see Boussinessq Problem, e.g., Malvern, 1969), or surface displacement (e.g., Farrell, 1972; Pinel et al., 2007). The case of induced

* Correspondence address: Berkeley Seismological Laboratory, University of California, Berkeley, 209 McCone Hall, Berkeley CA-94720, USA.
E-mail address: ronni@seismo.berkeley.edu
URLS: http://www.grapenthin.org/crusde, http://github.com/rgrapenthin/CrusDe

0098-3004/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cageo.2013.07.005

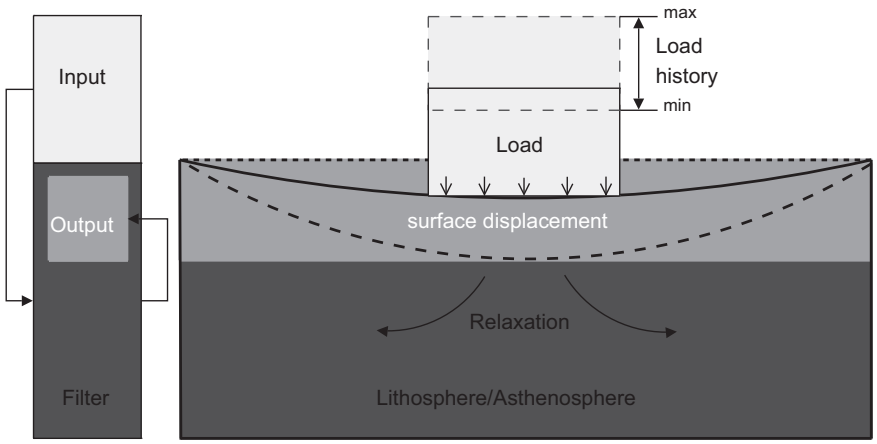

Fig. 1. Conceptual model of the Earth-load response system. The load (input, solid black rectangle) is applied to the pointed black line which represents the initial state of the Earth's surface. The Earth (filter) responds to the mass force (little black arrows) through surface displacement, changing its state to the one denoted by the solid black line (output). The load, however, may vary over time which is denoted by a load history. At first it might raise to a maximum level which results in maximum displacement and then it could drop to a minimum represented by the dashed gray lines that limit or extend the load box. Depending on the load history, the displacement might alternate between the upper and the lower dashed, gray surface lines, linked to minimum and maximum load, respectively. In addition to these elastic properties, another important property is that of time dependent stress relaxation or material creep which is the adjustment to a new stress state by ductile material flow, which in the depicted system is a filter property.

surface displacements is schematically shown in Fig. 1 and will be the focus of this paper.

The goal of surface load studies depends much on the problem at hand. Some are interested in the properties of the crust (e.g., Grapenthin et al., 2006; Pinel et al., 2007), the viscosity of the upper mantle (e.g., Pagli et al., 2007), or how the melting of ice caps can contribute to future volcanism (e.g. Jull and McKenzie, 1996; Pagli and Sigmundsson, 2008), and others want to forward model a known load, which can be used to isolate other signals in the data (e.g., Agnew, 1997; Grapenthin et al., 2010, in press). The basic principle for these applications remains the same: the conceptual model of the Earth–load response-system shown in Fig. 1 is translated into the mathematical framework of a convolution of a load with a Green's function, which is, in effect, the filtering operation described above.

Green's functions represent the unit impulse response of a system to an input (e.g., the surface will subside by X mm when a load of Y kg at distance Z km is applied; more see below and Roach, 1982). Considering that many Green's functions for different problems/Earth structures and a variety of problem domains exist, we should consider creating a tool that is heavily based on principles of software reuse. This would minimize the efforts of individual users when they try to solve their specific (loading) problem with a Green's function approach.

In this paper I present the architecture and implementation of CrusDe (installation information, download, and test cases at http://www.grapenthin.org/crusde), a simulation framework for composable crustal deformation studies written in C/C++ for Linux platforms. I derive a generalized formal model that utilizes Green's functions to express the response of the Earth's crust to a change in surface loads. The formal model translates into a data flow model which forms the basis of CrusDe's architecture. The implementation is validated through comparisons of modeled surface displacements due to a disk load to analytical solutions and a reference implementation.

This paper focuses (1) on the introduction of a flexible tool that can be used to solve a range of problems while using the same core functionality, and (2) on providing a blueprint on how to set up niche scientific tools that enable reusability which ultimately opens ways towards community tools rather than a collection of redundant case-based binaries.

2. Green's functions for surface loading

Green's functions are auxiliary functions that provide a particular method to solve boundary value problems (Roach, 1982). They can be understood as the unit response of a linear filter to a Dirac delta function, i.e. a force acting on a very small entity (e.g., a short time (unit impulse), or a point (unit point mass)). Under the assumption that the Earth acts as a linear, space invariant filter as described by, e.g., Grapenthin (2007, Section 3.3), Green's functions are frequently used in surface loading studies (e.g., Farrell, 1972; Agnew, 1997; Pinel et al., 2007).

This basic idea is illustrated in Fig. 2. We consider the dynamics of a point $\vec{r}$ (cylindrical coordinates) on the Earth's surface, which is displaced due to the application of a unit point mass at point $\vec{r'}$. The response at $\vec{r}$ due to the unit point mass $L(\vec{r'})$ is defined by the Green's function $G(\vec{r}, \vec{r'})$. A real load, however, covers an area instead of a single point. This can be abstracted as a grid of unit point masses acting on (ideally) equidistant points of the surface (on non-uniform grids, the term dS in Eq. (1) needs to be adjusted for each area fragment). Because we assume a linear system, we can integrate over this area, A, to sum the displacements, U, each unit point mass induces at point $\vec{r}$

$$U(\vec{r}) = \int_A G(\vec{r} - \vec{r'}) L(\vec{r'})\, dS$$

$$U = G ** L \tag{1}$$

where the double-asterisk notation denotes a 2D convolution (e.g., Meffert and Hochmuth, 2004). Since unit point masses can be zero in A, Eq. (1) implies that geometries of loads are not restricted to a particular shape other than restrictions imposed by the underlying grid size.

Each conceptual model of the Earth to be simulated (e.g., homogeneous elastic half-space, layered half-space, etc.) as well as the actual response to be modeled (displacement, stress change, etc.) requires the derivation of its own Green's function. A variety of Green's functions exists already (e.g., Longman, 1962; Farrell, 1972; Pinel et al., 2007); all of which could be used within the framework of Eq. (1). This, however, can be expanded to include load histories and relaxation functions.

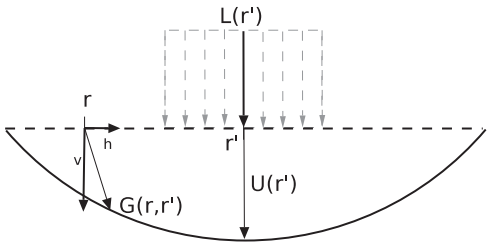

Fig. 2. Green's function for the response to an unit point mass. The displacement in point r due to a load $L(r')$ that is applied at r' is expressed by the Green's function $G(r,r')$. Since real data, e.g. from GPS stations, comes with displacements split into spatial directions, instead of calculating the total displacement it might be useful to find separate Green's functions for horizontal h and vertical v displacements. This is furthermore necessary since Green's functions are solutions to partial differential equations. The gray arrows represent a load model that is an array composed of many unit point masses, each having an impact on r. Utilizing Green's method (Eq. 1) and thus convolving the load with a Green's function, the impact of all unit point masses that compose the load is attributed to the displacement at r.

3. A generalized load response function

Green's functions, G^{∞} and G^{H}, respectively for the instantaneous and final relaxed response of the lithosphere to a surface load are derived by Pinel et al. (2007). These assume a flat elastic half-space Earth model or an elastic plate over an inviscid fluid, respectively. The notation G^{∞} and G^{H} for the Green's functions is adapted from Pinel et al. (2007) who use z to define the lower surface of the elastic plate: when $z=\infty$, we have a fully elastic medium and hence the instantaneous response G^{∞}; for the final relaxed response, G^{H}, Pinel et al. (2007) assume $z<\infty$, which defines a lower depth, H, of the elastic plate where hydrostatic fluid pressure is applied.

To realize the temporal evolution of displacements from the instantaneous to the final relaxed response, we introduce a function that describes the creep of ductile material. Analog to Pinel et al. (2007, Eq. 17) Eq. (1) can be expanded to

$$U_t = [(G^{\infty}-G^{H})**L]\cdot R_t + G^{H}**L \tag{2}$$

where U_t is the displacement at time t and R_t is the time dependent creep function, e.g. exponential decay (Pinel et al., 2007). The first half of the sum in Eq. (2) represents the viscous response of the lithosphere, which takes time to fully develop. The other half of Eq. (2) is the instantaneous elastic response of the crust, which has no time dependence.

For the case of time dependent load changes Pinel et al. (2007) give an expression for the displacement rate in their Eqs. (18) and (A3). Separation of variables for a time dependent load of the form $\Lambda(\vec{r},t)=L(\vec{r})H(t)$ must be possible to write

$$\dot{U}_t = [(G^{\infty}-G^{H})**L]\cdot(\dot{R}*\dot{H})_t + [G^{\infty}**L]\cdot(\delta*\dot{H})_t. \tag{3}$$

The requirement of separation of variables merely implies that each continuous "load chunk" has to obey the same load history function (initial values may vary). However, the assumption of linearity allows us to apply multiple loads at a time, each with their own load history, which mitigates the severity of the requirements posed by separation of variables.

In Eqs. (2) and (3) it is straightforward to substitute the term $G^{\infty}-G^{H}$ with G, a new Green's functions that expresses this sum. Since the convolution of a function with the δ−function is an identity operation, we can now give a general form for Eqs. (1)–(3) as

$$U_t = \sum_{i=0}^{n-1} [G_i**L]\cdot(R_i*H_i)_t \tag{4}$$

where n is the total number of superpositions of different applications of a Green's function that are necessary to find a solution for the respective problem (e.g., $n=2$ for Eqs. (2) and (3); $n=1$ for Eq. (1) with $R=H=\delta$).

In Eq. (4) the context determines whether time derivatives are to be used for all R_i and H_i, which yields the displacement rate $\dot{U}_t$ rather than the displacement U_t. Currently, the modeler has to ensure consistent use of time derivatives; future versions of CrusDe should add a safety mechanism. Note that the load L in Eq. (4) lacks the summation index since it defines the problem at hand and hence should not vary amongst the application of different Green's functions. Since the Earth–load response system is assumed to behave linearly, the convolution theorem holds and the convolutions in Eq. (4) can be performed as multiplications in the spectral domain which greatly enhances the performance of the operation (CrusDe's default convolution operator implements the convolution in the spectral domain; for details see e.g., Grapenthin, 2007, Section 3.6).

4. CrusDe's design

4.1. Conceptual outline

The basic idea behind CrusDe's architecture is in transferring the modularized structure of Eq. (4) into software, which we will refer to as a simulation framework. We see each operand and the convolution operator as abstract software components of a composable simulation model (Grapenthin, 2007, i.e., the user selects specific module realizations during runtime, which makes recompilation unnecessary). Fig. 3 maps the terms in Eq. (4) to software components in CrusDe and shows the data flow between them. The core of Eq. (4) is included for reference; its terms are vertically aligned with the corresponding software components shown in the boxes. The boxes in dark grey with bold arrows mark the minimum of participating software components; elements in light gray are optional. The mechanism realizing the summation in Eq. (4) is left out of the figure, but explained further below.

To realize the depicted data flow connecting the dark boxes, the software components implement role-specific interfaces that allow access to their data and operations. Conceptually speaking, a Green's function and a Load function will provide two dimensional matrices equal in size. The 2D convolution operator can request these matrices and convolve them. The result handler takes the result matrix for the current time step and writes it to the file system. Keeping the result handler component separate from the operator allows support of a range of (user defined) output formats. If desired, the result handler could implement an interface to third party software and hence enable CrusDe for re-use as a subsystem in more complex systems. The optional postprocessors can be used to transform results into other coordinate systems, compute ratios, perform statistical analyses, etc. The beauty of this approach lies in the total ignorance with which the convolution operator can approach the implementation details of its input functions. These functions only need to provide the required interfaces as specified (see Appendix A in supplementary data) and thus hide the complexity of their functionality from the system. Hence, one implementation of a convolution operator can be reused with a variety of input functions. This concept obviously results in software management overhead, which we detail in Section 4.2. Furthermore, the user who assembles the model must be intimately familiar with the functions used such that physically meaningful models are created.

Once we implement this mechanism which hides the complexity of individual components behind public interfaces, we create a

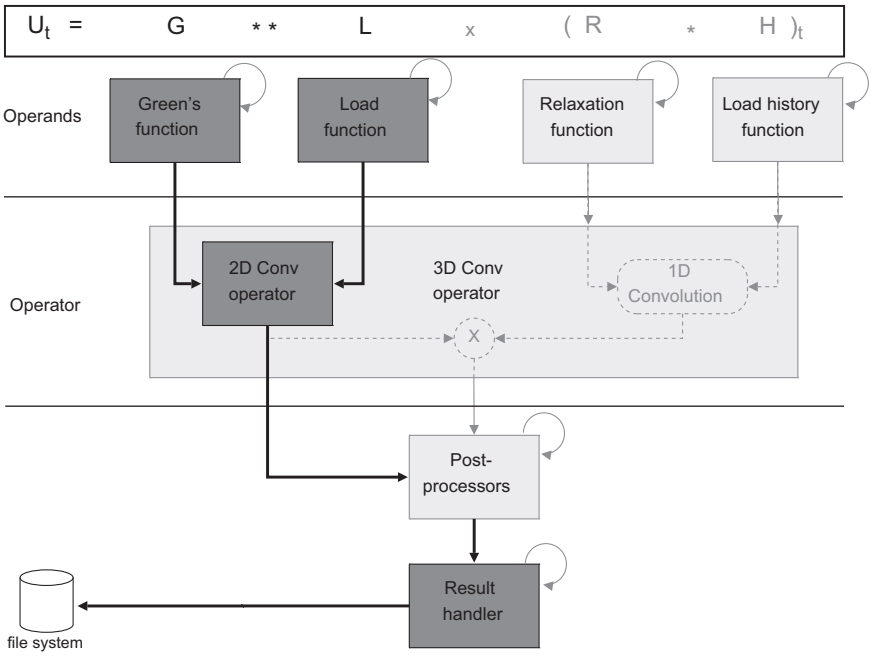

Fig. 3. Data flow (arrows) between important software components (boxes) of the proposed simulation framework. Since the framework is supposed to resemble the formal model (Green's method) the gray dotted lines denote the analogy between terms of Eq. (4) and software components. Additional software components are added to increase functionality and flexibility. Reflexive arrows refer to a software component that accesses functionality of another software component of the same category. The convolution operator takes data from the Green's and load function. The latter might be affected by a load history depending on whether or not a time dependent load is to be simulated. Once a convolution result for the examined area is calculated, one or more post-processors can be applied to the result. Finally, the results are passed to a result handler that writes the modeling results in a particular format to the file system. The terms in brackets denote references to sections where the respective component is detailed.

set of well-defined black-boxes[1] that enable software reusability. We can exploit this through reuse of existing components in new, more complex components (of the same category, see below). The reflexive arrows on the corners of the boxes in Fig. 3 indicate where this is allowed. The benefits of this are obvious when looking at, e.g., Pinel et al. (2007, Eqs. (2) and (9)) where Green's functions for an elastic half-space are reused in viscoelastic expressions.

Another example utilizing the reusability mechanism is displayed in the operator row of Fig. 3 where the 2D convolution operator is reused within the 3D convolution operator. Including a load history or a transition from instantaneous to final relaxed response into a simulation requires a convolution in space and time. Since convolutions of higher dimensions can be decomposed into multiple convolutions of lower dimension, the 3D convolution operator can be realized as shown by the dotted lines in Fig. 3. Therefore, an implementation of a 2D (spatial) convolution operator can be reused within a 3D (2D-space and time) convolution operator. The full implementation includes a 1D convolution of a creep function and a load history function, which is followed by a scalar multiplication of one element of the result vector with the 2D (spatial) convolution result. In cases which use just one of R or H, the 1D convolution can be skipped since this is the equivalent of a convolution with a $\delta-$function which yields the input function.

The approach outlined here brings advantages for the user, but it also introduces the problem of dependencies between software components. Page and Opper (1999) show that the decision whether an arbitrary collection of software components meets specified model requirements is NP-complete. To break down the complexity of resolving such dependencies, reuse within CrusDe is restricted to software components of the same category as indicated by rectangular boxes in Fig. 3. Categories form proper subsets of the overall set of available software components. Therefore, the dependencies of each software component are reduced to its own category and the respective required functions must be implemented as these define the category. Furthermore, this restriction greatly lowers the risk of mistakes on the user side by applying functions for tasks they were not intended for. On the downside, multiple implementations of identical functions in different categories could be a side effect and would ruin the overall idea of reuse. However, the nature of each of the categories indicates proper subsets of tasks that are very different from each other. The intersection between these subsets should be the empty set, which indicates minimal risk of redundant implementations.

4.2. Software architecture

Following the concept outlined above (Fig. 3), the requirements on the software architecture are such that a user can (i) dynamically select software components that participate in a simulation, (ii) extend the framework independently with new software components, and (iii) exchange software components and experiment definitions with peers.

To fulfill these requirements, the data flow in Fig. 3 is transferred to the three layer structure in Fig. 4 which represents CrusDe's architecture. The software components from Fig. 3 are included in the functional layer in Fig. 4 and implemented as plug-ins (shared libraries that are dynamically loaded during runtime, see Appendix B). *All* elements of the formal model (i.e., Eq. (4)) are contained in this layer. They are, however, not directly connected. Interface and management layer implement the data flow between components of the functional layer and thus implement the composable simulation model.

The central element of CrusDe is the Simulation core, which defines the management layer together with an Input handler, an Experiment manager and a Plug-in manager. The Simulation core

[1] "Black-box" is meant in the software sense, which means the calling function/module does not have to care about the individual implementation details. The user has to know these details to create a physically meaningful model or sensible new components. CrusDe makes all these details available in source code and documentation.

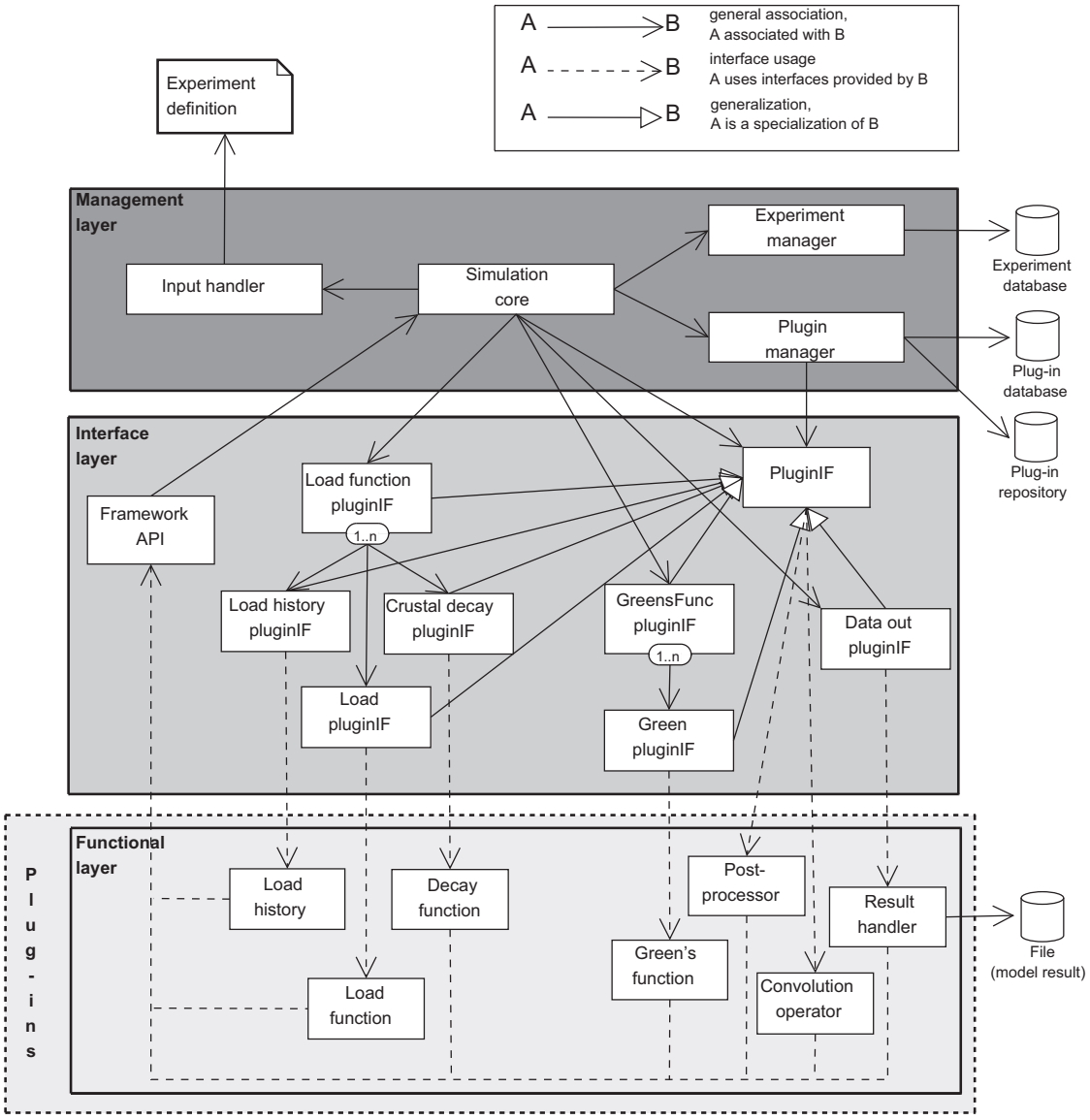

Fig. 4. Architecture of the proposed plug-in based simulation framework. The software components of Fig. 3 are represented by plug-ins of the functional layer. The logical data flow depicted in Fig. 3 is realized by the interface and management layers which provide the infrastructure to support communication between the plug-ins. Input handler, experiment manager, and plug-in manager provide additional functionality to implement the plug-in concept. The experiment definition contains the configuration of a simulation.

implements the main loop which specifies the sequence in which plug-ins and management elements operate. Furthermore, the Simulation core manages function calls invoked by plug-ins at CrusDe's provided-interface, the Framework API. The Simulation core will route the call to the respective needed-interface which calls the related required function the corresponding plug-in has to implement.

The Input handler interprets the experiment definition (e.g., Listing 1), which specifies a simulation run in an Extensible Markup Language (XML) file. Plug-ins participating in this simulation, their parameter settings, and additional information such as the coordinates of the region of interest as well as spatial and temporal resolution are defined in this file. The Input handler validates the experiment definition to a certain degree (i.e., ensure the defined format is followed) and makes the contents accessible to the Simulation core in an appropriate form (e.g., convert parameter values from string literals to numbers).

The Experiment manager keeps a history of the individual simulations run through CrusDe. Metadata, such as parameter settings, modeling result location, used plug-ins, user, time, and date are stored in a database (XML) to allow a later matching of modeling results to experiments or reconstruction of experiments.

The Plug-in manager has access to a plug-in repository (directory hierarchy) and a plug-in database (XML). The plug-in repository is the location in the file system where the shared libraries are stored in a controlled and organized way upon registration with CrusDe. Metadata must be provided by each plug-in (through special functions, see Appendix A) such that it can be stored in an XML database which is used to locate the respective binary when used for a simulation. Values stored in the database are plug-in name, textual description of its functionality, its category, required parameters, requested plug-ins, and its location in the plug-in repository. This is in accord with the demands of Overstreet et al. (2002) who identify the capturing of objectives, assumptions, and constraints as a key to reuse. However, before adding a plug-in to the system, the Plug-in manager checks whether all the functions required for the specified category are implemented and whether dependencies are fulfilled, i.e., whether the required plug-ins are installed.

When the Simulation core retrieves the physical location of a plug-in through the Plug-in manager, it creates instances of their needed-

```
<?xml version="1.0" encoding="UTF-8"?>
<experiment name="disk">
  <file name="result" value="./disk_elastic_high.nc" />
                                                <!-- result file -->
  <region name="west"   value="-20000" />       <!-- region of interest, Lambert coords [m] -->
  <region name="east"   value="20000"  />
  <region name="south"  value="-20000" />
  <region name="north"  value="20000"  />

  <parameter name="timesteps" value="12"/>      <!-- 12 time steps (e.g. days) -->
  <parameter name="gridsize"  value="500"/>     <!-- 500x500 m grid cells -->

<!-- We use time, therefore 3D convolution! -->
 <kernel> <plugin name="fast_3d_convolution" /> </kernel>

<!-- Elastic Halfspace, Pinel et al. (2007) -->
 <greens_function>
    <plugin name="elastic_halfspace_(pinel)"/> <!-- plugin name as in plugin database -->
    <parameter name="g" value="9.81" />         <!-- acceleration due to gravity [m s^-2] -->
    <parameter name="nu" value="0.25" />        <!-- poisson ratio -->
    <parameter    name="E" value="10"/>           <!-- Youngs modulus [GPa] -->
 </greens_function>

<!-- Disk load in the center of the model region that changes height following a  -->
<!-- sinusoidal function. Note that 'load' and 'load_history' are both defined in -->
<!-- load function.                                                               -->
 <load_function>
    <load>
       <plugin name="disk_load" />
       <parameter name="height" value="150"/>   <!-- load height [m] -->
       <parameter name="radius" value="4000" /> <!-- disk radius [m] -->
       <parameter name="center_x" value="0" />  <!-- x location -->
       <parameter name="center_y" value="0" />  <!-- y location of disk center -->
       <parameter name="rho" value="1000" />    <!-- density [kg m^-3] -->
    </load>

  <!--twice the timesteps to simulate only load build-up, peaks on the simulated day -->
    <load_history>
       <plugin    name="sinusoidal" />
       <parameter name="period_length" value="24" />
       <parameter name="peak" value="12" />
    </load_history>
  </load_function>

<!-- Post-processor here converts displacements in x-y direction to radial -->
<!-- This could contain multiple post-processors.                           -->
  <postprocessor> <plugin name="xy2r" /> </postprocessor>

<!-- The output writer writes the results in netCDF format to the file specified above. -->
  <output> <plugin name="netcdf_writer"/> </output>
</experiment>
```

Listing 1: Experiment Definition: Disk load with load history on an elastic half-space.

interfaces which are implemented through `PluginIF` or one of its derived classes in the Interface layer. These objects bind the respective shared libraries, which makes their functionality accessible to CrusDe. Fig. 4 shows in the Interface layer which interface binds instances of which plug-in. The main difference in the individual *IF implementations is the parameterization of the 'run'-function which defines the core functionality of a plug-in. A load function plug-in, for instance, implements the calculation of the point mass at a given location of the modeled area in its 'run'-function. It needs coordinates as parameters. A post-processor, however, is simply told to work; without any parameters, it requests its inputs through CrusDe's API.

Functions that access plug-in metadata are inherited[2] from the `PluginIF`-interface. Loaded plug-ins use the Framework API to request data from other plug-ins or allocate resources from CrusDe.

The architecture of CrusDe must furthermore account for two additional important features that are associated with the

[2] In object-oriented design an inheritance relation describes, here at the example of interfaces, that an interface A that inherits interface B contains B (e.g. all its functions) as a subset. Thus, at runtime instances of interface A can be interpreted as both interface A and (with the limited functionality of B) interface B.

Simulation core in the Management layer (these features are not included in Fig. 3 to limit its complexity):

- The functionality of a plug-in may depend on an unknown number of parameters (defined by the plug-in) which get their values for each simulation assigned in the experiment definition. Thus, a *parameter registry* is implemented and associated with the Simulation core to provide a link between definition and initialization of the parameters. Every plug-in can register an arbitrary number of parameters. Before a simulation starts the Input handler assures that the user provided all necessary values and no parameters remain uninitialized in order to get each plug-in to operate.
- A *plug-in registry* is implemented to allow individual plug-ins to load and access the functionality provided by another plug-in of its category.

For more detail, Grapenthin (2007) visualizes initialization sequence, plug-in communication, and execution flow in the form of UML sequence diagrams.

4.3. Job mechanism

A peculiarity in Fig. 3 are the '1…n' boxes attached to the `LoadFunctionPluginIF` and `GreensFuncPluginIF` interfaces. This illustrates that these interfaces can hold multiple load functions and Green's functions, respectively, which is a way to realize the sum in Eq. (4). A 'job'-mechanism enables the computation of crustal responses defined by different Green's functions (see Listing 2). Each term of the sum can be defined as a job within the Green's function specification in the experiment definition. The Simulation core will hand a list of jobs to the `GreensFuncPluginIF`-interface which manages the individual Green's functions. Within its main loop the Simulation core keeps track of the current job. The summation of the individual job results is realized in the 3D convolution operator.

As the user is also allowed to specify multiple load functions in the experiment definition, the `LoadFunctionPluginIF` manages the respective load, load history, and decay functions. It holds '1…n' instances of the associated plug-ins (for load history and crustal decay plug-in interfaces '0…n' is valid as Eq. (4) and therefore the experiment definition requires only a load to be

```
[... Lines 0-14 from Listing 1, set 'timesteps' to 1 though...]

<!-- Elastic Halfspace final response, equation 17 in Pinel et al. (2007) -->
  <greens_function>
   <!-- define multiple jobs to make use of superposition, job labels can be -->
   <!-- used within load_function, see below                                  -->
   <job name="thickplate">
     <plugin name="inverse_thick_plate_(pinel)"/> <!-- Greens function A-->
   </job>

   <job name="final_relaxed">
      <plugin name="final_relaxed_(pinel)"/>      <!-- Greens function B; result is A+B -->
   </job>

   <!-- list ALL requested parameters -->
   <parameter name="g" value="9.81" />          <!-- used before -->
   <parameter name="nu" value="0.25" />      <!-- used before -->
   <parameter name="E" value="10"/>          <!-- used before -->
   <parameter name="H" value="5000"/>        <!-- NEW: elastic plate thickness [m] -->
   <parameter name="rho_f" value="3100"/>       <!-- NEW: inviscid fluid density [kg m^-3] -->
  </greens_function>

  <!-- same disk as in previous example, no load history though -->
  <load_function>
    <load>
       <plugin name="disk_load" />
       <parameter name="height" value="150"/>   <!-- load height [m] -->
       <parameter name="radius" value="4000" /> <!-- disk radius [m] -->
       <parameter name="center_x" value="0" />  <!-- x location -->
       <parameter name="center_y" value="0" />  <!-- y location of disk center -->
       <parameter name="rho" value="1000" />    <!-- density [kg m^-3] -->
    </load>

    <!-- crustal decay will only be applied to "thickplate" Greens function-->
    <crustal_decay job="thickplate">
     <plugin     name="exponential" />           <!-- exponential decay -->
     <parameter name="tR" value="300" />         <!-- Maxwell relax. time [yrs] -->
    </crustal_decay>

  </load_function>

[... Lines 44-50 from Listing 1 ...]
```

Listing 2: Abbreviated Experiment Definition: Showing the `job' mechanism that allows for superposition of multiple Green's functions (e.g., Pinel et al., 2007, Eq. (17))

defined). The Simulation core keeps track of the current load component that is being worked with.

Crustal decay and Load histories can be defined for specific 'jobs.' However, as of now, a Load function definition can hold only a single load history and one crustal decay definition. Those can be linked to either a specific job or all jobs. Allowing for multiple load history or crustal decay definitions within a single load function, each linked to different jobs, may be included in future versions, but this demands significant code changes.

5. Testing and validation: disk load

Maximum vertical and minimum (i.e., zero) horizontal displacement are obtained under the center of a disk. An analytical solution for the vertical displacement at this point derived from Eq. (1) is given by Geirsson et al. (2006)

$$U_{v,center} = 2\rho h R_0 g \frac{1-\nu^2}{E} \quad (5)$$

where $U_{v,center}$ is the vertical displacement under the center of a disk, ρ is the density of the load, h is the load height, R_0 is the radius of the disk, g is the acceleration due to gravity, and ν and E are, respectively, Poisson's ratio and Young's modulus.

Table 1 shows the results for a disk of height $h=150$ m, radius $R_0=2$ km, and density $\rho=1000$ kg m^{-3} applied to an elastic half-space with a Young's modulus of $E=10$ GPa and a Poisson's ratio of $\nu=0.25$. We compare CrusDe's results to the analytical solution and results of a similar simulation model that implements the same load models and Green's functions for the elastic half-space, but performs the convolution in the spatial rather than the frequency domain (Grapenthin and Sigmundsson, 2006). The full spatial solution from CrusDe's run is given in Fig. 5.

CrusDe misses the analytical solution by 2.2 mm in the vertical and 0.1 mm in the horizontal displacement. However, the differences in Table 1 can be explained by machine precision, different techniques for the convolution, and discretization of the model domain (for details see Grapenthin and Sigmundsson, 2006, Table 5.1). Comparing the displacements shown in Fig. 5a and b to the solution of Grapenthin and Sigmundsson (2006) shows identical patterns. The horizontal displacement is maximum in the area of the disk edge and vanishes under its center because of the load symmetry and the uniform layering of the model (Fig. 5e). Finally, conforming to Pinel et al. (2007) the ratio between horizontal and vertical displacements does not exceed $1/3$ for an elastic half-space with $\nu=0.25$ (Fig. 5c and f, Table 1).

We do not show results for simulations that utilize irregular loads and a sinusoidal load history. However, these conform to results obtained by Grapenthin et al. (2006). We also observe that with increasing plate thickness the response of the thick plate model equals the response of the elastic half-space as stated by Pinel et al. (2007).

Table 1
Comparison of CrusDe's solution for the response under the center of a disc load to a reference implementation and an analytical solution. The disk is characterized by height $h=150$ m, radius $R_0=2$ km, and density $\rho=1000$ kg m^{-3} and its center is at $x=y=5000$ m. The elastic half-space has a Young's modulus of $E=10$ GPa and a Poisson's ratio of $\nu=0.25$. The grid size is 10×10 m for a $10\,000\times10\,000$ m region of interest.

	Analytical solution	Reference implementation[a]	CrusDe's solution
$U_{v,center}$ (m)	0.5518[b]	0.5500	0.5496
$\lvert U_{h,center}\rvert$ (m)	0	0	0.0001
$\lvert U_h\rvert/U_z$	$\leq 1/3$[c]	$\leq 1/3$	$\leq 1/3$[d]

[a] See Grapenthin and Sigmundsson (2006).
[b] From Eq. (5).
[c] Note that this must be true for the whole modeled area, not just the center, see Pinel et al. (2007).
[d] See Fig. 5.

6. Discussion

The implementation of a code like CrusDe that enables reuse of components and defines experiments removed from the actual code base on a more abstract level clearly requires more effort on

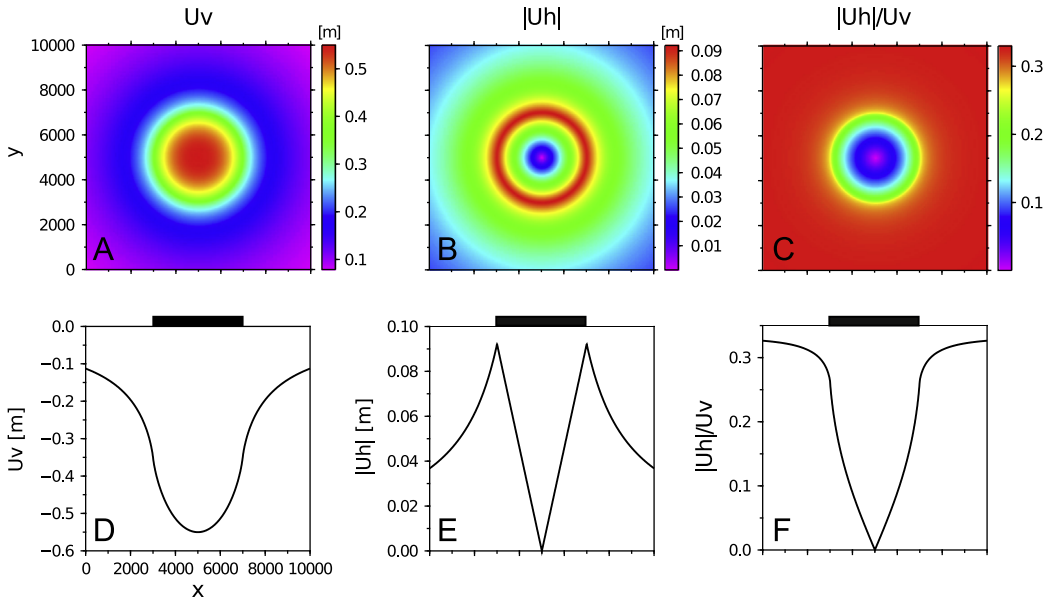

Fig. 5. Response of an elastic half-space to a disk load. (A) Vertical displacement, (B) horizontal displacement, (C) ratio of horizontal and vertical displacements. (D–F) Same as (A–C) but in a cross-section through $y=5000$ m; the black rectangle shows the horizontal extent of the disk, vertical is not to scale. The Young's modulus and Poisson's ratio of the half-space are set to 10 GPa and 0.25, respectively. The disks parameters are set to height $h=150$ m, radius $R_0=2$ km, and density $\rho=1000$ kg m^{-3}. Its center is at $x=5000$ m and $y=5000$ m.

the developer side. This effort goes mainly into the definition of interface and management layers to allow for data flow between the individual components, as well as handling input (interpretation of experiment definition) and output. This is opposed to the effort required by more straight-forward implementations, in which code is produced for one particular model, likely with hard coded parameter values. For the latter to work, a tremendous bookkeeping effort is required to keep software versions and modeling results synchronized, and results reproducible once new models are to be tested or different parameter values need to be checked. The approach introduced here, however, presents one robust implementation of the core mechanism, i.e. convolution, that can be expanded as needed. This reduces the efforts of everybody using the Green's function approach.

Tests with a disk load demonstrate that the infrastructure works and CrusDe produces results similar to the analytical solution and the reference implementation. Results obtained by Grapenthin et al. (2006) and Pinel et al. (2007) can be reproduced. The diversity of applications for CrusDe is reflected in previous studies it has been successfully applied to, such as forecast of subsidence due to a water reservoir (Ófeigsson et al., 2006), removal of lava load signals from deformation data (Grapenthin et al., 2010; Ófeigsson et al., 2011), and estimation of isostatic adjustment due to a volcanic cluster (Grapenthin et al., in press).

A drawback of CrusDe's current design is its reliance on metric measurements for distances on the grid. This results in issues with Green's functions for a spherical Earth (e.g., Farrell, 1972), which usually give distances in degrees, and it limits the ability to reuse existing code. Agnew (2012) for example provides an implementation of spherical Earth Green's functions (Farrell, 1972). Reusing their code with minimal changes is not possible unless a translation from metric distances to degrees was performed. This results in unnecessary complexity at any time degree based operations are required. A future version of CrusDe will include this in the Simulation handler.

The currently provided Green's functions also are not necessarily implemented in the computationally most efficient way. We use analytic expressions which means that the Green's function is calculated for each model run. A more efficient way would be to use pre-computed lookup tables (e.g., Farrell, 1972) and interpolate between values. Given CrusDe's architecture, this would be straightforward to do as it only requires to generate the tables and write a new plug-in that reads the Green's function from a given file. Future versions may include this.

Lastly, CrusDe is setup such that results are given for only one level of depth (current Green's functions give displacement at the surface only). If Green's functions that give output for multiple depth levels are used, they should be parameterized such that the depth level can be set in the experiment definition. A wrapper script that generates experiment definitions with different depth values and calls CrusDe multiple times would be straightforward to implement.

7. Conclusions and outlook

This paper provides a blueprint showing how to set up niche scientific tools to enable software reusability. Ultimately, this opens ways towards community tools rather than a collection of redundant, case-based binaries.

We derived a generalized load response function (Eq. 4) that includes load history, crustal relaxation, Earth model and load model functions. This is implemented in a plug-in based simulation framework (implemented and tested in C/C++ on Linux platforms) which enables users to switch simulation components based on XML definitions without duplicating large parts of the code base. CrusDe comes with a range of plug-ins ready to be used for simulations of surface displacements Appendix B in supplementary data. The current Green's functions give surface displacements for the elastic, final relaxed, pure thick plate response of a flat Earth to surface load changes. These can be combined to express the exponential decay from elastic to final relaxed response, and displacement rates (see Pinel et al., 2007) due to one or multiple disks, or irregular loads, or a combination of these. Each of these load functions can have its own load history and crustal decay functions. Expansions to this or simplifications (precomputing Green's functions and interpolation of look-up tables) are straightforward to add to the tool as new plug-ins. The code has been tested against analytical solutions for disk loads and results that were obtained from independent implementations of the same Green's functions.

Future work on CrusDe will focus on direct support of spherical Earth models as well as the integration of additional Green's functions. Following this, an interface to ALMA (Spada, 2008) is planned, which will enable the derivation of Green's functions from layered Earth models through the summation of Love numbers (e.g., Farrell, 1972).

Acknowledgments

I want to thank the editor Jef Caers as well as Maurizio Battaglia and an anonymous reviewer for constructive comments and very efficient handling of the manuscript. Ideas for this work emerged through collaboration with Freysteinn Sigmundsson during a visit at NordVulk—the Nordic Volcanological Center, Institute of Earth Sciences, University of Iceland. Most of the work presented here was accomplished as part of my M.Sc. work at the Humboldt University in Berlin, Germany, which was guided and supported by my advisors Joachim Fischer and Freysteinn Sigmundsson. Much of the refinement of CrusDe was implemented early during my Ph.D. at the University of Alaska Fairbanks. Thankfully, my Ph.D. advisor Jeff Freymueller let me continue to work on CrusDe during this time and provided support through the Alaska Volcano Observatory. I am also indebted to Virginie Pinel who helped me greatly with the details of Pinel et al. (2007). Matt Groening helped by developing that colorful character which inspired the name CrusDe.

Appendix. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org.10.1016/j.cageo.2013.07.005.

References

Agnew, D.C., 1997. NLOADF: a program for computing ocean-tide loading. Journal of Geophysical Research 102, 5109–5110.
Agnew, D.C., 2012. SPOTL: some programs for ocean-tide loading. Technical Report. Scripps Institution of Oceanography. La Jolla.
Farrell, W., 1972. Deformation of the Earth by surface loads. Reviews of Geophysics 10, 761–797.
Frigo, M., Johnson, S.G., 2005. The design and implementation of FFTW3. In: Proceedings of the IEEE, vol. 93. pp. 216–231.
Geirsson, H., Árnadóttir, T., Völksen, C., Jiang, W., Sturkell, E., Villemin, T., Einarsson, P., Sigmundsson, F., Stefánsson, R., 2006. Current plate movements across the Mid-Atlantic Ridge determined from 5 years of continuous GPS measurements in Iceland. Journal of Geophysical Research 111, B09407.
Grapenthin, R., 2007. CrusDe: A Plug-in Based Simulation Framework for Composable CRUStal DEformation Simulations using Green's Functions. Master's Thesis. Berlin.
Grapenthin, R., Freymueller, J.T., Serovetnikov, S.S. Surface deformation of Bezymianny volcano, Kamchatka, recorded by GPS: the eruptions from 2005–2010 and long-term, long-wavelength subsidence. Journal of Volcanology and Geothermal Research, in press.

Grapenthin, R., Ófeigsson, B.G., Sigmundsson, F., Sturkell, E., Hooper, A., 2010. Pressure sources versus surface loads: analyzing volcano deformation signal composition with an application to Hekla volcano, Iceland. Geophysical Research Letters 37, 3–7.

Grapenthin, R., Sigmundsson, F., 2006. The Green's functions technique in crustal deformation and its applications. Technical Report. The Nordic Volcanological Center, Reykjavik, Iceland.

Grapenthin, R., Sigmundsson, F., Geirsson, H., Árnadóttir, T., Pinel, V., 2006. Icelandic rhythmics: annual modulation of land elevation and plate spreading by snow load. Geophysical Research Letters 33, 1–5.

Jull, M., McKenzie, D., 1996. The effect of deglaciation on mantle melting beneath Iceland. Journal of Geophysical Research 101, 21,815–21,828.

Longman, I., 1962. A Green's function for determining the deformation of the Earth under surface mass loads. Journal of Geophysical Research 67, 845–850.

Malvern, L., 1969. Introduction to the Mechanics of a Continuous Medium. Prentice-Hall, New Jersey.

Meffert, B., Hochmuth, O., 2004. Werkzeuge der Signalverarbeitung. Pearson Studium.

Ófeigsson, B., Einarsson, P., Sigmundsson, F., Sturkell, E., Ólafsson, H., Grapenthin, R., Geirsson, H., 2006. Expected crustal movements due to the planned Halslón reservoir in Iceland. In: Eos Transactions AGU, Fall Meeting Supplement, pp. Abstract T13A–0495.

Ófeigsson, B.G., Hooper, A., Sigmundsson, F., Sturkell, E., Grapenthin, R., 2011. Deep magma storage at Hekla volcano, Iceland, revealed by InSAR time series analysis. Journal of Geophysical Research 116, 1–15.

Overstreet, C.M., Nance, R.E., Balci, O., 2002. Issues in enhancing model reuse. In: Overstreet, C.M., Nance, R.E., Balci, O. (Eds.), Issues in Enhancing Model Reuse. International Conference on Grand Challenges for Modeling and Simulation, January 27–31, San Antonio, Texas, USA, San Antonio, Texas, USA.

Page, E., Opper, J., 1999. Observations on the complexity of composable simulation. In: Proceedings of the 1999 Winter Simulation Conference, pp. 553–560.

Pagli, C., Sigmundsson, F., 2008. Will present day glacier retreat increase volcanic activity? Stress induced by recent glacier retreat and its effect on magmatism at the Vatnajökull ice cap, Iceland. Geophysical Research Letters 35, L09304.

Pagli, C., Sigmundsson, F., Lund, B., Sturkell, E., Geirsson, H., Einarsson, P., Árnadóttir, T., Hreinsdóttir, S., 2007. Glacio-isostatic deformation around the Vatnajökull ice cap, Iceland, induced by recent climate warming: GPS observations and finite element modeling 112, B08405.

Pinel, V., Sigmundsson, F., Sturkell, E., Geirsson, H., Einarsson, P., Gudmundsson, M.T., Högnadóttir, T., 2007. Discriminating volcano deformation due to magma movements and variable surface loads: application to Katla subglacial volcano, Iceland. Geophysical Journal International 169, 325–338.

Roach, G., 1982. Green's Functions, 2nd ed. Cambridge University Press, Cambridge.

Spada, G., 2008. ALMA, a Fortran program for computing the viscoelastic Love numbers of a spherically symmetric planet. Computers & Geosciences 34, 667–687.

Watts, A.B., 2001. Isostasy and Flexure of the Lithosphere. Cambridge University Press.